24. (New) The computer-readable medium of claim 11, wherein:

tag potions of the tagged numeric reference comprise N bits and store at least a first tag value and a second value indicating complementary properties; and

a difference of the first tag value and the second tag value is congruent to $2^{N-1}$ modulo $2^N$.

25. (Once Amended) The method of claim 11 wherein the sum consists of the tagged numeric reference and the second tagged machine pointer.

26. (New) The computer-readable medium of claim 15, wherein:

the memory is subdivided into a plurality of pages;

the first object is stored on a first page; and

the second object is stored on a second page, other than the first page.

## REMARKS

By this amendment, claims 1-26 are pending, in which claim 25 is amended. No new matter is introduced.

The final Office Action mailed April 7, 2003 objected to claims 4, 10, 14, 20-21, and 24 and rejected claims 5 and 15 under 35 U.S.C. § 102 as anticipated by *Lee* (US 6,345,276), claims 1-2, 6-8, 11-12, and 16-18 as obvious under 35 U.S.C. § 103 based on *Lee* (US 6,345,276) in view of *Murray* (Kelly E. *Murray*, "Under the Hood: CLOS"), and claims 3, 9, 13, and 19 as obvious under 35 U.S.C. § 103 over *Lee* and *Murray* further in view of *Carter et al.* (US 6,003,123).

In response to the objections of claims 22 and 25, claim 25 has been amended so that it no longer depends on independent claim 1.

The rejection of claims 5 and 15 is respectfully traversed because *Lee* alone or in combination with *Murray* fails to disclose the limitations of the claims. For example, independent claims 5 and 15 recite:

> **storing** a reference within a first object to a second object in the memory as a numeric reference that **encodes a location of the second object** as an **offset** from an address of the first object in the memory

This limitation is not found in *Lee*. Specifically, with reference to FIG. 2, object **203** at offset 20 in the heap **201** contains a smart pointer **204** to object **207** at offset 300 in the heap **201**. The Examiner correctly calculates that the "the difference of fields 205 and 206 (300-20=**280**) is the **offset** of object1 207 from the address of object0 203." (Page 7, lines 7-9, emphasis added) However, this offset value of 280 is plainly not stored anywhere in FIG. 8 as a reference as required by this limitation or anywhere else. Smart pointer **204** actually stores the value 20 in portion **205** (the offset of the offspring pointer **204** from the start of the heap **201**), and the value 300 in a destination-pointer portion **206** (the offset of the object **207** in the heap **201**). Even though the difference can be computed from these fields of the smart pointer, that difference is not stored in the smart pointer. Therefore, *Lee* does not disclose "**storing** a reference within a first object to a second object in the memory as a numeric reference that **encodes a location of the second object** as an **offset** from an address of the first object in the memory."

With respect to the rejections of claims 1-2, 6-8, 11-12, and 16-18 under § 103, there is no motivation to combine *Lee* and *Murray* to produce an invention having the following element: "generating the first tagged machine pointer as a **sum** including the tagged numeric reference and the second tagged machine pointer."

The Office Action responded that "*Lee* does not appear to require objects to be of different types." (Page 7, last line) If true, then this invalidates the Office Action's stated motivation on pp. 4-5 to combine *Lee* with *Murray* in the first place: "It would have been obvious the pointers as taught by *Lee* would also be tagged because the tag values of a pointer can provide important information such as distinguishing between pointers and non-pointers. (Column 3, first paragraph, p. 82 of *Murray*.)" The cited passage of *Murray* states: "Type information plays a central role in Lisp implementation. It is used by the garbage collector to distinguish between pointers and non-pointers, and is also required to support heterogeneous collections and polymorphism." (Page 82, column 3)

On the other hand, if *Lee* permits objects to be of different types, then modifying *Lee* to use the pointer tags of *Murray* to encode type information as *Murray* discloses would not result either in the claimed invention or in an operable embodiment. Specifically, use of *Murray*'s tag values do not work in the *Lee* system because the address calculation of *Lee* if combined with *Murray* would produce a pointer to a destination object that are tagged with the type of the wrong object. In other words, the pointer to a destination object would be tagged with the type of the source object. For example, in *Lee*, the address of object **207** is calculated in two steps, **317** and **319**. In step **317**, a base pointer is calculated by subtracting offset **205** from pointer **203**. Since offset **205** is a number, its tag according to *Murray* would be 00, and the base pointer would therefore have the same tag as pointer **203**. In step **319**, the base pointer is added to the destination offset **206** to produce what should be, but is not, the tagged pointer to object **207**. Since the destination offset **206** is also a number, its tag is 00 and the result of step **319** is a pointer with the same tag as the base pointer, which has the same tag as the pointer to object **203**. However, object **207** can have a different type from that of object **203**, so the object **203**'s tag is wrong with respect to object **207**'s type. In fact, there is no assignment of type tags in *Murray*

that would result in the generated pointer to object **207** always having the right type tag. Accordingly, combining *Lee* with *Murray* results in an inoperable system in which all destination pointers are tagged with the type of the source pointers.

The remaining dependent claims 2-3, 6-9, 13-19, 23-23, and 25-26 are allowable for at least the same reasons as their independent claims and are individually patentable on their own merits. For example, even though *Carter et al.* discloses a virtual memory system, there is no value in a tag portion that "indicates whether the first object has a same or a different contiguity as a contiguity of the second object" as recited in claims 3, 9, 13, and 19. Rather, *Carter et al.* at best shows two different virtual page identifiers, in two different Global Translation Lookaside Buffer (GTLB) entries, which only indicate the number of the virtual page (cols. 17:62-67 and 18:15-23). Merely knowing the virtual page numbers is not enough to know whether the first or second object is contiguous or discontiguous or whether the contiguity they have is the same or different.

The Office Action takes the position that "if two object references have the same virtual page identifier, then they would belong in the same page group and have the same contiguity." (Page 6, item 7). However, it is unclear how belonging to the same page group has anything to do with having the same contiguity, and the Office Action gives no explanation of continuity that is consistent with what a person of skill in the art would understand it to mean. "When not defined by applicant in the specification, the words of a claim must be given their plain meaning. In other words, they must be read as they would be interpreted by those of ordinary skill in the art." (MPEP 2111.01)

Therefore, the present application, as amended, overcomes the objections and rejections of record and is in condition for allowance. Favorable consideration is respectfully requested. If any unresolved issues remain, it is respectfully requested that the Examiner telephone the

undersigned attorney at 703-425-8516 so that such issues may be resolved as expeditiously as possible.

Respectfully Submitted,

DITTHAVONG & CARLSON, P.C.

5/22/03
Date

Leila R. Abdi
Attorney/Agent for Applicant(s)
Reg. No. 52399

10507 Braddock Rd
Suite A
Fairfax, VA 22032
Tel. 703-425-6499
Fax. 703-425-8518

## APPENDIX

25. (Once Amended) The method of claim [1] 11 wherein the sum consists of the tagged numeric reference and the second tagged machine pointer.